S P E C I F I C A T I O N

TO ALL WHOM IT MAY CONCERN:

Be it known that I, Henrique S. Malvar, a citizen of
Brazil, residing at 2302 233rd Avenue NE, Sammamish, Washington
98074, have invented a certain new and useful **IMPROVED BLOCK
TRANSFORM AND QUANTIZATION FOR IMAGE AND VIDEO CODING** of which
the following is a specification.

IMPROVED BLOCK TRANSFORM AND QUANTIZATION
FOR IMAGE AND VIDEO CODING

FIELD OF THE INVENTION

5   The invention relates generally to the encoding and

decoding of image and video signals, and more particularly to

an improved block transform and inverse transform, along with

quantization and de-quantization methods, for encoding and

decoding image video signals.

10

BACKGROUND OF THE INVENTION

        The Discrete Cosine Transform (DCT) is commonly used in

block transform image and video (sequences of images) coding,

e.g., JPEG and MPEG. Among other reasons, this is because the

15 DCT separates random signals (such as corresponding to image

data) into low frequency parts of greater importance and high

frequency parts of lesser importance with respect to the

image's visual quality, thus facilitating compression.

        By way of example, H.26L is an image and/or video codec

20 technology that constructs frames using four-by-four blocks of

pixels. Unlike MPEG and JPEG, which uses eight-by-eight

blocks, H.26L obtains relatively high-quality compression with

four-by-four blocks of pixels by using prediction information

of existing other blocks in the same frame, (i.e., intra-frame

25 coding), in addition to estimation and motion compensation

between frames, (i.e., inter-frame coding). In general, to

accomplish intra-frame prediction, an H.26L encoder tells the corresponding H.26L decoder what other, previous block of pixels in the frame being built is being used as a basis for the prediction, along with identifying which one of six

5   possible predictors (formulas) to use in determining the pixels for the new block (from those of the previous intra-frame block). This leads to a prediction error, which is also provided to the decoder to correct the new block. The prediction error information is encoded with the block

10   transform (DCT) and sent encoded to the decoder, for re-computing the error information, including via an inverse transform, to correct the predicted block.

Image and video encoding or decoding takes a significant amount of processing power. As is known, an encoder typically

15   has far more processing power than do many of the decoders that convert the data to images, as decoders are typically implemented in consumer devices. For example, image and video decoding may take place in television set-top boxes, personal digital assistants (PDAs), pocket-sized personal computers, and

20   more advanced cellular telephones.

Thus, when considering image and video encoding and decoding methods, keeping the decoding simple is important, even if it means that encoding has to be made more computationally complex. Anything that can simplify decoding

25   is desirable, provided that in doing so, the encoding and

decoding processes do not adversely impact existing levels of compression, final image quality and the amount of other resources needed at the decoder in any substantial way relative to existing technology.

5

SUMMARY OF THE INVENTION

Briefly, the present invention provides an improved method, system and block transform that significantly simplifies computational complexity for images and video at

10 both the encoder and decoder. At the same time, compression, image / video quality, and other resources are only negligibly affected.

More particularly, an orthogonal block transform and corresponding inverse transform with new integer approximations

15 to the Discrete Cosine Transform (DCT) are provided, along with other computing changes that significantly reduce computational complexity at both the encoder and decoder. Indeed, in one implementation, at the encoder and decoder, the number of transform operations, per coefficient, has been reduced to four

20 additions and one shift in 16-bit arithmetic, (from four additions and three multiplications in 32-bit arithmetic required with the H.26L-specified transform). The present invention correctly transforms the pixel information (e.g., error correction data) because quantization (scaling and

25 rounding to an integer) during encoding, and de-quantization

- 3 -

during decoding, via the use of one of three tables selected based on each coefficient's position, have parameter values that already compensate for factors of other transformation multiplications, except for the one performed by the shift operation during the transformation and inverse transformation processes.

Further, during decoding, on each quantized coefficient, the decoder can perform a sixteen-bit multiplication to de-quantize that quantized coefficient into a transform coefficient, instead of a thirty-two bit multiplication (as required in H26.L). On many devices, with typical images and video, the processing benefits of the present invention are thus significant relative to H.26L, especially at the decoder, with only negligible if any loss in quality and/or compression.

Other benefits and advantages will become apparent from the following detailed description when taken in conjunction with the drawings, in which:

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1 is a block diagram representing an exemplary computer system into which the present invention may be incorporated;

FIG. 2 is a block diagram representing components used in encoding pixels of video or image data into encoded video, and decoding the encoded video into pixel information, including

transformation, quantization, de-quantization and inverse

transformation in accordance with one aspect of the present

invention;

FIG. 3 is a representation of a butterfly structure that

represents the improved transformation matrix in accordance

with one aspect of the present invention; and

FIG. 4 is a representation of a butterfly structure that

represents the improved inverse transformation matrix in

accordance with one aspect of the present invention.

DETAILED DESCRIPTION

EXEMPLARY OPERATING ENVIRONMENT

FIGURE 1 illustrates an example of a suitable operating

environment 120 in which the invention may be implemented,

particularly for decoding image and/or video data. The

operating environment 120 is only one example of a suitable

operating environment and is not intended to suggest any

limitation as to the scope of use or functionality of the

invention. Other well known computing systems, environments,

and/or configurations that may be suitable for use with the

invention include, but are not limited to, personal computers,

server computers, hand-held or laptop devices, multiprocessor

systems, microprocessor-based systems, programmable consumer

electronics, network PCs, minicomputers, mainframe computers,

distributed computing environments that include any of the

above systems or devices, and the like. For example, it is likely that encoding image and/or video image data often will be performed on a computer with more processing power than contemporary hand-held personal computers, but there is no

5 reason encoding cannot be performed on the exemplary device, or decoding on a more powerful machine.

The invention may be described in the general context of computer-executable instructions, such as program modules, executed by one or more computers or other devices. Generally,

10 program modules include routines, programs, objects, components, data structures and so forth that perform particular tasks or implement particular abstract data types. Typically the functionality of the program modules may be combined or distributed as desired in various embodiments.

15 Computing device 120 typically includes at least some form of computer readable media. Computer-readable media can be any available media that can be accessed by the computing device 120. By way of example, and not limitation, computer readable media may comprise computer storage media and communication

20 media. Computer storage media includes volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is

25 not limited to, RAM, ROM, EEPROM, flash memory or other memory

technology, CD-ROM, digital versatile disks (DVD) or other

optical storage, magnetic cassettes, magnetic tape, magnetic

disk storage or other magnetic storage devices, or any other

medium which can be used to store the desired information and

5   which can accessed by the computing device 120. Communication

media typically embodies computer readable instructions, data

structures, program modules or other data in a modulated data

signal such as a carrier wave or other transport mechanism and

includes any information delivery media. The term "modulated

10   data signal" means a signal that has one or more of its

characteristics set or changed in such a manner as to encode

information in the signal. By way of example, and not

limitation, communication media includes wired media such as a

wired network or direct-wired connection, and wireless media

15   such as acoustic, RF, infrared and other wireless media.

Combinations of the any of the above should also be included

within the scope of computer readable media.

FIG. 1 shows functional components of one such handheld

computing device 120, including a processor 122, a memory 124,

20   a display 126, and a keyboard 128 (which may be a physical or

virtual keyboard). The memory 124 generally includes both

volatile memory (e.g., RAM) and non-volatile memory (e.g., ROM,

PCMCIA cards, and so forth). An operating system 130 is

resident in the memory 124 and executes on the processor 122,

such as the Windows® CE operating system from Microsoft®
Corporation, or another operating system.

One or more application programs 132 are loaded into
memory 124 and run on the operating system 130. Examples of
applications include email programs, scheduling programs, PIM
(personal information management) programs, word processing
programs, spreadsheet programs, Internet browser programs, and
so forth. The handheld personal computer 120 may also include
a notification manager 134 loaded in the memory 124, which
executes on the processor 122. The notification manager 134
handles notification requests, e.g., from the application
programs 132.

The handheld personal computer 120 has a power supply 136,
which is implemented as one or more batteries. The power
supply 136 may further include an external power source that
overrides or recharges the built-in batteries, such as an AC
adapter or a powered docking cradle.

The exemplary handheld personal computer 120 represented
in FIG. 1 is shown with three types of external notification
mechanisms: one or more light emitting diodes (LEDs) 140 and an
audio generator 144. These devices may be directly coupled to
the power supply 136 so that when activated, they remain on for
a duration dictated by a notification mechanism even though the
handheld personal computer processor 122 and other components
might shut down to conserve battery power. The LED 140

preferably remains on indefinitely until the user takes action.
Note that contemporary versions of the audio generator 144 use
too much power for today's handheld personal computer
batteries, and so it is configured to turn off when the rest of

5   the system does or at some finite duration after activation.

ENCODING AND DECODING

By way of background, the Discrete Cosine Transform (DCT)
maps a length-N vector **x** into a new vector **X** of transform

10   coefficients by a linear transformation **X** = **H x**, where the
element in the kth row and nth column of **H** is defined by

$$\mathbf{H}_{kn} = H(k,n) = c_k \sqrt{\frac{2}{N}} \cos\left((n+1/2)\frac{k\pi}{N} \right)$$

15   for k = 0, 1, …, N-1, and n = 0, 1, …, N-1, with $c_0 = \sqrt{2}$ and $c_k = 1$
for k > 1. The DCT matrix is orthogonal, so its inverse equals
its transpose, that is **x** = **H**$^{-1}$ **X** = **H**^{T} **X**.

One disadvantage of the DCT is that the entries H(k,n) are
irrational numbers, and so integer input data x(n) will map to

20   irrational transform coefficients X(k). As a result, with
digital computers, when the direct and inverse transforms are
computed in cascade, the output data does not exactly equal the
input data. In other words, if computed as **X** = **H x**, and **u** =
round(**H**^{T} **X**), then it is not true that u(n) = x(n) for all n.

- 9 -

However, by introducing appropriate scale factors α, γ, e.g., $X = \gamma\ H\ x$ and $u = \text{round}(\alpha\ H^T\ X)$, then $u(n) = G\ x(n)$, where G is an integer, for almost all n, when α and γ are chosen to be large enough, but this does not guarantee an exact result.

5   In a motion-compensated video encoder, for example, the data of past decoded frames are used as reference information for prediction information that will be used to generate the current frame. Therefore, as part of encoding, the encoder generates its own decoded frames, whereby the encoder needs to

10   compute inverse transforms. If the formula $u = \text{round}(\alpha\ H^T\ X)$ is used, then different floating-point formats and rounding strategies in different processors lead to different results. As a result, there will be a drift between the decoded data at the encoder versus that decoded by decoders (which have

15   different processors), whereby the image worsens more and more per each new frame, since the encoder is basing prediction / motion information on frames of blocks that are increasingly less and less like the frames of blocks that the decoder is producing.

20   One solution to the data drift problem approximates the matrix H by a matrix containing only integers, whereby rounding errors are eliminated. If the rows of H are orthogonal and have the same norm (sum of the squares), then it follows that u can be computed exactly in integer arithmetic for all integer

25   x. In other words, when the direct transform is computed by X

= **H** **x** and the inverse transform by **u** = **H**^{T} **X**, then **u** = G **x**

results, where G is an integer equal to the squared norm of any

of the rows in **H**.

One way to generate integer approximations to the DCT is

5   by using the general formula:

$$Q(k,n) = \text{round } (\alpha\ H(k,n))$$

where α is a scaling parameter.

In the H.26L video compression standard, the image is made

up of blocks of four pixels by four pixels, whereby N = 4 in

10   the DCT formula in H.26L.   This results in the DCT matrix

equaling:

$$\mathbf{H} = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ c & s & -s & -c \\ 1 & -1 & -1 & 1 \\ s & -c & c & -s \end{bmatrix}$$

where $c \triangleq \sqrt{2}\cos(\pi/8)$ and $s \triangleq \sqrt{2}\sin(\pi/8)$.

15   The transform matrix in the current version of H.26L is

obtained by setting α = 26, whereby the values compute to:

$$\mathbf{Q}_0 = \text{round}(26\mathbf{H}) = \begin{bmatrix} 13 & 13 & 13 & 13 \\ 17 & 7 & -7 & -17 \\ 13 & -13 & -13 & 13 \\ 7 & -17 & 17 & -7 \end{bmatrix}$$

With a scaling parameter value of 26, the rows and columns

20   of $\mathbf{Q}_0$ are orthogonal to each other (i.e., the inner product of

- 11 -

any two columns is zero), and all rows and columns have a norm equal to 676. In fact, for values wherein $\alpha < 50$, only $\alpha = 2$ or $\alpha = 26$ provide orthogonal matrices with equal norm rows. However, the solution for $\alpha = 2$ does not lead to good compression, and larger values for α are not attractive because of the increase in the computational complexity (e.g., word length) required to compute the results of the direct transform $\mathbf{X} = \mathbf{Q_0}\ \mathbf{x}$. Hence, $\alpha = 26$ has been heretofore chosen in H.26L.

The inverse transform is defined by $\mathbf{x'} = \mathbf{Q_0}\ \mathbf{X}$, so that it can also be computed with integer arithmetic. From the definition above, $\mathbf{x'} = 676\ \mathbf{x}$, i.e. the reconstructed data $\mathbf{x'}$ is equal to the original data $\mathbf{x}$ amplified by an integer gain of 676 (which is the norm of any of the rows in $\mathbf{Q_0}$).

In accordance with one aspect of the present invention, a block transform matrix is provided that uses integer approximations that are orthogonal, yet have different values that significantly reduce computational complexity when encoding and decoding. More particularly, using $\alpha = 2.5$, the following matrix is generated:

$$\mathbf{Q_D} \triangleq \text{round}(2.5\mathbf{H}) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

Note that the rows of Q_D are orthogonal to each other, and, while their norms are different (rows zero and two have norms equal to four, while rows one and three have norms equal to ten), as described below this is handled in quantization and

5   de-quantization in a manner that does not significantly increase computational complexity.

Turning to FIG. 2 of the drawings, there is shown a block transform encoder 200 and block transform decoder 222, connectable via by some type of medium and/or intermediate

10 devices 220. As is understood, an encoder 200 need not make its output directly available to the decoder 222, and thus the line labeled 220 in FIG. 2 may represent virtually any type of medium or devices, such as a network cable, wireless transmission medium, telephone line, storage device, router,

15 and/or virtually any combination thereof.

In general, the encoder 200 operates by transforming each block of NxN input pixel data 202 (e.g., representing error correction information) via a separable two-dimensional transforms 204. More particularly, first the rows of pixel

20 data within the block are transformed by a row transformation process 206, and then the results of the row transformation (not specifically represented in FIG. 2) are transformed by a column transformation process 208 into the resulting transform coefficients 210. In other words, the two-dimensional DCT

25 essentially performs a one-dimensional DCT on each row of a

block of pixels, followed by a one-dimensional DCT on each
column of the block of pixels that was produced by the one-
dimensional DCTs on the rows. The row and column
transformations can be performed in reversed order, with the

5   same result obtained.

In H.26L, the specified transform matrix results in the
following formulas for computing the transform coefficients:

$$A = 13a + 13b + 13c + 13d$$
$$B = 17a + 7b - 7c - 17d$$
$$C = 13a - 13b - 13c + 13d$$
$$D = 7a - 17b + 17c - 7d$$

10

where [abcd] first represents a row of pixel values, and then,
after row transformation, represents a column of those row-

15   transformed data values, in a four-by-four block. However, in
practice, these formulas can be simplified, and the DCT matrix
has a recursive structure that reduces the number of required
multiplication and addition operations. Nevertheless, the
above matrix requires at least four additions and three

20   multiplications to compute the transform coefficients for each
pixel. Moreover, because each pixel may be a signed nine-bit
value, when multiplied by the row and column factors (the gain
equals the norm of 676) the computation requires 32-bit
arithmetic, in which a single computation takes as much time as

25   many 16-bit computations. While these considerations are not
normally as significant during encoding, with these H.26L
specified matrix values, the extra operations and 32-bit

- 14 -

arithmetic also occur during decoding, wherein the expense is
significant.

In contrast to the H.26L-specified matrix, using the same
general formula representation, as described above the present
5   invention's block transform matrix is:

$$\mathbf{Q_D} \triangleq \text{round}(2.5\mathbf{H}) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

which provides the following formulas for computing the
transform coefficients:

```
A = a  +  b +  c +  d
B = 2a +  b -  c - 2d
C = a  -  b -  c +  d
D = a  - 2b + 2c -  d
```

Part of the reason that this formula / matrix works more
15   optimally is that instead of performing individual
multiplications in the transformation stage, the multiplication
factors are essentially handled in the quantization phase, by
simply changing the values used in scaling. As a result, the
only multiplication with these formulas is by a power of two,
20   which in processors is accomplished via a simple shift left
operation, not an actual multiplication. With u, v, y and z
used as auxiliary variables, and wherein "<< 1" means shift
left one bit (equivalent to multiplying by two but
computationally faster), the above formulas simplify to the
25   following formulas:

- 15 -

```
u = a + d;
v = b + c;
y = b - c;
z = a - d;
A = u + v;
C = u - v;
B = y + (z << 1);
D = z - (y << 1);
```

Via the butterfly structure represented in FIG. 3,
(wherein a line between two points indicates a sum, and the
numbers on the lines (the -1s, 2 and -2 indicate a
multiplication factor) finding these transform coefficients
actually requires only four additions and one shift for each
given pixel, providing a very substantial performance
improvement relative to the H.26L-specified factors. Moreover,
because of the lower coefficient values, all transformation
operations may be performed in 16-bit arithmetic (with 9-bit
pixel data). More particularly, with the transform matrix Q_D
defined above, after a two-dimensional 2-D transform, the
maximum signal amplification is 36, whereby the output
coefficients after the two-dimensional transform will span 6
more bits than the input. Therefore, for 9-bit input, the
output coefficients have a dynamic range of 15 bits, meaning
that the two-dimensional direct transform can be computed with
16-bit arithmetic. Moreover, as described below, the
performance increase resulting from far fewer operations, each
of which requires only 16-bit arithmetic, is essentially

- 16 -

repeated during decoding operations with the inverse transform, providing a significant increase in decoding performance.

Once the transform coefficients 210 are computed, the block of coefficients 210 is quantized by scaling the values and rounding them to their nearest integers. This is represented in FIG. 2 by the quantization process 212, which selects from among quantization values from quantization parameters 214 to scale the transform coefficients 210 into quantized coefficients 216. As described above, because there is not one norm, but three in the improved matrix, three quantization tables Q0, Q1 and Q2 contain these values, depending on where the transformed coefficient to be quantized is positioned in the block.

The quantization formula for scaling and rounding is as follows, which should be computed with 32-bit precision:

$$L = [K \times A(QP,r) + fX] >> 20$$

where L is the quantized coefficient, K is the transformed coefficient, $A(QP,r)$ is the scale factor indexed by the quantization parameter QP and r, wherein r identifies which table (Q0, Q1 or Q2 to use), fX is in the range $[0-0.5] \times 2^{20}$ (fX has the same sign as K), and >> 20 means shift right twenty places (divide by 1,048,576), to scale the number down. The quantization process thus introduces errors. Note that although 32-bit arithmetic is used for quantization, this is only required when encoding, which is not as significant as

when decoding (wherein de-quantization needs only 16-bit

precision, as described below).

The index r thus selects which of the three quantization

tables Q0, Q1 or Q2 to use to adjust for the three different

5   norms, based on the position of the coefficient K in the block:

r=0 (use Q0) if the coefficient came from one of the
positions {(0,0),(0,1),(1,0),(1,1)};

r=1 (use Q1) if the coefficient came from one of the
10   positions
{(0,2),(0,3),(1,2),(1,3),(2,0),(2,1),(3,0),(3,1)};

r=2 (use Q2) if the coefficient came from one of the
positions {(2,2),(2,3),(3,2),(3,3)}.

15

After the right shift by 20 bits, each quantized result L fits

into a 16-bit integer.

In an exemplary implementation, the scaling factor A(QP,

20   r) depends on the quantization parameter QP and coefficient

position group r according to the following tables:

A(QP=0..31, r=0) = {104858, 93418, 83226, 74146,
66056, 58849, 52429, 46709, 41613, 37073, 33028,
25   29425, 26214, 23354, 20806, 18536, 16514, 14712,
13107, 11677, 10403, 9268, 8257, 7356, 6554, 5839,
5202, 4634, 4129, 3678, 3277, 2919};

A(QP=0..31, r=1) = {66318, 59082, 52636, 46894,
30   41778, 37220, 33159, 29541, 26318, 23447, 20889,
18610, 16579, 14771, 13159, 11723, 10444, 9305, 8290,
7385, 6580, 5862, 5222, 4652, 4145, 3693, 3290, 2931,
2611, 2326, 2072, 1846};

35   A(QP=0..31, r=2) = {41943, 37367, 33290, 29658,
26422, 23540, 20972, 18684, 16645, 14829, 13211,
11770, 10486, 9342, 8323, 7415, 6606, 5885, 5243,
4671, 4161, 3707, 3303, 2942, 2621, 2335, 2081, 1854,
1651, 1471, 1311, 1168}.

40

- 18 -

The particular values above were designed to satisfy the design specification in H.26L that the quantization step sizes should double for every increment of six in the table index. For other video or still image coding applications, other

5   quantization tables can be designed, in view of the desired fidelity increments. Note that to store the quantization tables at the encoder, some extra table space is needed to store three tables instead of one, but this amount of extra space is only 64 bytes, which is negligible.

10   Returning to FIG. 2, following the quantization process, the quantized coefficients 216 are fed to an entropy encoder 218 which, in general, further reduces the number of bits needed to encode the block. Entropy encoders (and decoders) are known, and are thus not described herein, as the present

15   invention operates before and essentially independent of entropy encoding, and after and essentially independent of entropy decoding.

Turning to a consideration of decoding in the block transform decoder 222, at some time, regardless of how

20   delivered, the entropy encoded output bits are fed as input bits to an entropy decoder 224. In general, such entropy decoders are known, and thus among other possible operations, it suffices to state that the entropy encoder 224 reproduces the quantized coefficients 226 for a given block. Note that

25   typically the encoding and decoding are not lossy, i.e., the

quantized coefficients 216 fed into the entropy encoder 218 will be identical to the quantized coefficients 226 produced by entropy decoder 224.

In general, the block transform decoder 222 mirrors the operation of the block transform encoder 200, although as described below and in keeping with the present invention, a modified inverse transform matrix is provided that enables 16-bit arithmetic to be used throughout the decoding process, thus greatly simplifying the decoder's computational complexity.

Once the quantized coefficients 226 are retrieved from the entropy decoder 224, each quantized coefficient L is converted to a reconstructed (de-quantized) value K' by a de-quantization process 228 implementing the formula:

$$K' = L \times B(QP, r)$$

where the scaling factor B depends on the QP index used when encoding, and r determines the de-quantization parameters 230 by selecting a parameter from one of the tables D0, D1, or D2 (to adjust for the three norms). Note that r can be deduced from the position of the coefficient being de-quantized in the block, as described above with respect to quantization, although it is feasible to send the r value from the encoder, which would possibly reduce computational complexity on the decoder but would require at least two more bits be sent per coefficient to identify r.

In this manner, the scaling factor B(QP, r) depends on the

quantization parameter index for QP used when encoding, and the

coefficient position group r. In an exemplary implementation,

that dependency is specified according to the following tables:

B(QP=0..31, r=0) = {80, 90, 101, 113, 127, 143, 160, 180, 202, 226, 254, 285, 320, 359, 403, 453, 508, 570, 640, 718, 806, 905, 1016, 1140, 1280, 1437, 1613, 1810, 2032, 2281, 2560, 2874};

B(QP=0..31, r=1) = {101, 114, 127, 143, 161, 180, 202, 227, 255, 286, 321, 361, 405, 454, 510, 572, 643, 721, 810, 909, 1020, 1145, 1285, 1443, 1619, 1817, 2040, 2290, 2570, 2885, 3239, 3635};

B(QP=0..31, r=2) = {128, 144, 161, 181, 203, 228, 256, 287, 323, 362, 406, 456, 512, 575, 645, 724, 813, 912, 1024, 1149, 1290, 1448, 1625, 1825, 2048, 2299, 2580, 2896, 3252, 3650, 4095, 4596}.

As with the encoding tables, the particular values above were

designed to satisfy the design specification in H.26L that the

quantization step sizes should double for every increment of

six in the table index. For other video or still image coding

applications, other quantization tables can be designed, in

view of the desired fidelity increments.

As can be appreciated, the de-quantization formula scales

the coefficients back up to larger values, although each of

these scaling values is low enough to ensure that only 16-bit

arithmetic is needed at the decoder. It should be noted that

since each entry in a table is mathematically related (based on

the norm) to corresponding QP-indexed table entries in the

other two tables, it is alternatively possible have only one

table of QP values, with an appropriate mathematical adjustment based on the r value, rather than have three tables. However, lookups in small tables are relatively efficient, and the number of bytes required to store three tables of thirty-two

5   entries at two bytes per entry instead of one table of thirty-two entries at four bits per entry (required with the specified H.26L transform) is sixty-four extra bytes, which is negligible in contemporary computing devices, especially considering that the transform coefficients generated by de-quantization fit

10   into sixteen bit words instead of 32-bit DWords, thereby reducing the total amount of memory needed.

As also represented in FIG. 2, once the quantized coefficients 226 have been de-quantized via the above formula and tables, a four-by-four matrix of reconstructed transform

15   coefficients 232 is present. From those transform coefficients 232, the reconstructed pixels are generated by feeding them to a two-dimensional inverse transform process 234 comprising column and row inverse transform processes 238 and 238, respectively.

20   In keeping with the present invention, instead of using the inverse transform $Q_I = Q_D^T$, as would normally be done, to allow for 16-bit de-quantization and inverse transform computation, the present invention defines the following inverse transform matrix for use:

$$\mathbf{Q_I} \triangleq \begin{bmatrix} 1 & 1 & 1 & 1/2 \\ 1 & 1/2 & -1 & -1 \\ 1 & -1/2 & -1 & 1 \\ 1 & -1 & 1 & -1/2 \end{bmatrix}$$

Note that the columns of Q_I are orthogonal to each other, but their norms are different. However, this was handled via the "r" value used in selecting the de-quantization parameter, as

5   described above. Further, note that in generating Q_I from Q_D, Q_D was transposed, with columns one and three multiplied by one-half. Again, however, the QP values already present in the tables D0, D1 and D2 have been adjusted for this in advance, eliminating the need for any additional computational

10   complexity to compensate. Moreover, the only "multiplication" that will need to be done when computing the inverse transforms with such a matrix is by one-half, which is actually performed by a highly-efficient shift right operation. Note that the small amount of noise introduced by such imprecise division has

15   essentially no impact in the rate-distortion performance.

With Q_I defined as above, its maximum gain equals four. Therefore, the two-dimensional inverse transformation process only expands the dynamic range by four bits, allowing for computation in 16-bit arithmetic.

20   The column inverse transform process 236 and row inverse transform process 238 (that operates on the result of the column inverse transform process) represented in FIG. 2 generate the output pixel data 240. To this end, using the

above inverse transform matrix and calling [A B C D] a

particular row or column to be inverse transformed, the

corresponding inverse transformed values [a' b' c' d'] are

computed by:

5

```
u = A + C;
v = A - C;
y = (B >> 1) - D;
z = (D >> 1) + B;
a' = u + z;
b' = v + y;
c' = v - y;
d' = u - z;
```

10

15   where u,v,y and z are auxiliary variables and >>1 means shift

one bit to the right (equivalent to multiplying by one-half).

Similar to encoding, via the butterfly structure represented in

FIG. 4, these equations actually reduce to four addition

operations and one shift operation per coefficient, all in 16-

20   bit arithmetic. Lastly, the inverse transform process 234

(FIG. 2) scales the reconstructed pixel values by a factor of

2^{-7} (performed via a 7-bit shift-right operation).

The reconstructed pixel values after the 7-bit shift have

a 9-bit range, so the inverse transform can be computed in 16-

25   bit arithmetic. The de-quantization tables D0, D1 and D2 set

forth above are such that the de-quantization equations will

not overflow when computed in 16-bit arithmetic. The following

tables summarizes some of the computational advantages when

using the transform and inverse transform of the present

30   invention relative to those specified in H.26L:

- 24 -

Transform	Operations per pixel, direct transform	Quant-ization	De-quant-ization	Operations per pixel, inverse transform
H.26L Specified	4 additions, 3 multiplications in 32-bit arithmetic	1 mult. in 32-bit arithmetic	1 mult. in 32-bit arithmetic	4 additions, 3 multiplications in 32-bit arithmetic
Present Invention	4 additions, 1 shift, in 16-bit arithmetic	1 mult. in 32-bit arithmetic	1 mult. in 16-bit arithmetic	4 additions, 1 shift, in 16-bit arithmetic

As can be readily appreciated, such savings are significant, particularly on 16-bit capable processors. For example, in

5 practice, a typical 16-bit PDA processor may see an improvement in decoding of roughly double the speed with the present invention.

Moreover, the transform / inverse transform have been tested with respect to its impact on quality and compression.

10 Results for coding gain (usually defined as an increase in signal-to-noise ratio) show that the use of the transforms of the present invention with test data results in a loss of only about 0.01 dB relative to the H26L transforms, which is negligible, and in fact is likely far smaller than that with

15 actual video signal information, such as pixel prediction errors. Further, overall performance was tested relative to the H.26L specified transforms across each of the thirty-two quantization parameters, resulting in only a plus or minus half percent range of differences, with an average very close to

20 zero, thereby indicating that the transform of the present

- 25 -

invention performed as well as the transform specified in
H.26L.

As can be seen from the foregoing detailed description,
there is provided an improved transform and inverse transform
5   matrices for image or video encoding and decoding,
respectively, that significantly reduce computational
complexity with respect to other known transforms without
adversely impacting compression or quality. Transformation
multiplications are eliminated yet correct results are obtained
10   because quantization and de-quantization parameter values
compensate for those multiplication factors, except for one
performed by a shift operation during the transformation and
inverse transformation processes. Because of the values used,
transformation operations during encoding, and de-quantization
15   and transformation operations during decoding can be performed
in 16-bit arithmetic (for pixel data represented by nine bits
or less).

While the invention is susceptible to various
modifications and alternative constructions, certain
20   illustrated embodiments thereof are shown in the drawings and
have been described above in detail. It should be understood,
however, that there is no intention to limit the invention to
the specific forms disclosed, but on the contrary, the
intention is to cover all modifications, alternative

constructions, and equivalents falling within the spirit and scope of the invention.